**Q1 how javascript work internally?**

**Q2 single thread execution?**

**Q3 global context?**

**Q4 execution context?**

**Color = green 🡪 code Examples.**

**Color = Red 🡪 important topic and concepts.**

**1. What is React.js**

React.js is an open-source JavaScript library (created by Facebook) used to build interactive user interfaces, especially for single-page applications (SPAs). It lets you break your UI into reusable components and updates the page efficiently when data changes.

Think of it as a smart system that updates only what’s necessary on a page instead of reloading the whole page.

**2. Why you should learn React**

* High demand – Many companies (including big tech) use React, so it increases job opportunities.
* Reusable components – Write once, reuse multiple times, making development faster.
* Fast rendering – Uses a virtual DOM for quick updates.
* Large ecosystem – Many libraries, tools, and community support.
* Cross-platform – You can use React Native to build mobile apps with the same skills.

**3. Usage of React**

React is used for:

* Web applications (e.g., Facebook, Instagram, Netflix)
* Dashboards and admin panels
* E-commerce websites
* Interactive forms and user inputs
* Single-page apps where content changes without full reload

**4. What to learn before React**

Before diving into React, you should have a strong foundation in:

1. HTML
   * Elements, attributes, forms, semantic tags.
2. CSS
   * Styling, positioning (flexbox, grid), responsive design.
3. JavaScript (ES6 and above)
   * Variables (let, const), data types, arrays, objects.
   * Functions (normal and arrow functions).
   * DOM manipulation (querySelector, addEventListener).
   * Events and event handling.
   * Template literals and string methods.
   * Array methods (map, filter, forEach, reduce).
   * Object destructuring and spread/rest operators.
   * this keyword basics.
   * Conditional statements and loops.
4. Basic Git/GitHub (optional but very useful)
   * Version control and pushing projects online.

**why the component name must start with a capital letter in React.**

React uses **capitalization** to know whether something in JSX is a **component** or a plain HTML element.

* If it **starts with lowercase** (like <div> or <span>), React treats it as a built-in HTML tag.
* If it **starts with uppercase** (like <First> or <Random>), React treats it as a **custom component**.

So, if you write this:

function random() {

return <h1>Hello</h1>;

}

export default random;

and then render <random />, React will think "random" is an HTML tag, not your component, so it won’t call your function.

**Correct way:**

function Random() {

return <h1>Hello</h1>;

}

export default Random;

and then:

import Random from "./First.jsx";

function App() {

return <Random />;

}

Basically — React sees <First> or <Random> and says “Ah, that’s a component I should run” but <first> or <random> looks like a strange HTML element and gets ignored.

**How creat elemnt using react?**

**<></> // fragment**

**Code:**

const textApp = "hi every one"; // variable

const anotherReactElement = React.createElement(

  "a",

  { href: "https://google.com", target: "\_blank" },

  "Visit google !",

  textApp // inject variable it’s a evaluated expression

);

**✅ What is a Hook in React?**

**Hooks** are special functions in React that let you **use state, lifecycle, and other React features** in **functional components**.

🔥 Before hooks, only class components could use things like state and componentDidMount. But now with hooks, you can do all that in **functional components** (which are simpler and cleaner).

**🤔 Why Do We Use Hooks?**

We use hooks to:

* Add **state** to functional components
* Run code when components **mount, update, or unmount**
* **Reuse logic** without changing the component structure
* Write **cleaner, modern React code**

**🔹 Common Hooks in React:**

| **Hook** | **Purpose** |
| --- | --- |
| useState | To manage component state |
| useEffect | To perform side effects (API calls, etc.) |
| useContext | To use global values like themes |

**✅ What is useState Hook?**

useState is a React Hook that lets you **add and update state** in functional components.

**🔧 Syntax:**

const [state, setState] = useState(initialValue);

* state: current value of the state
* setState: function to update that value
* initialValue: default value (e.g., "", 0, false, [], etc.)

**🧪 Example: Counter App**

import { useState } from 'react';

function Counter() {

const [count, setCount] = useState(0); // initial value is 0

return (

<div>

<h2>Count: {count}</h2>

<button onClick={() => setCount(count + 1)}>Increment</button>

</div>

);

}

**🧠 What’s Happening?**

* useState(0) → sets count to 0 initially
* setCount(count + 1) → updates count when button is clicked
* React re-renders the component with the **new value**

**🎯 Summary:**

* useState allows **state management** in functional components.
* It replaces this.state and this.setState from class components.
* Hooks like useState make React code **simpler, shorter, and cleaner**.

**Documentation**

1.The createRoot create's its own DOM and then compare it with the web browser's DOM and only update those components which are actually updated.

2.But the browser removes the whole DOM and then recreates the whole DOM with the updated values this is called reload.

3. However virtual DOM tracks whole DOM like a tree like structure and updates only those values which were only changed.

4. But some values depends on network call so if we update a value it might get update immediately via a network call.

5. So we will have to update it again. To avoid this overhead we can drop the updation calls for the immediate value update.

6. The current algo used by the React is called the React Fibre algo.

7. The algo react uses to differentiate the web browser's tree and React's tree formed through create root is called reconciliation.

8. Reconciliation is the algo behind what popularly known as the Virtual-DOM.

9.In UI it is not necessary for every update to be applied immediately.

**Hydration:**

hydration : Jab pehli baar page load hota hai, buttons aur images dikhte hain (HTML aa gaya), lekin kuch click nahi hota kyunki JavaScript ab tak load nahi hui hoti. Phir jab JavaScript load ho ke React ko HTML ke saath connect karti hai aur sab interactive ho jata hai, us process ko hydration bolte hain. aur ye fibre algorithm k through kaafi aachi hoti hai.

**Props:**

In React, **props** (short for *properties*) are a way to pass data from a parent component to a child component.

They are similar to function parameters, and the child component receives them as an object.

**Key points about props in React:**

1. **Read-only** – A component cannot change its own props.
2. **Used for communication** – They let parent components send data, event handlers, or configuration to child components.
3. **Passed like HTML attributes** – When using a component, you write props as attributes inside the JSX tag.

**Example:**

// Child Component

function Greeting(props) {

return <h1>Hello, {props.name}!</h1>;

}

// Parent Component

function App() {

return <Greeting name="Bilal" />;

}

export default App;

**Explanation:**

* name="Bilal" is a prop passed from App (parent) to Greeting (child).
* Inside Greeting, we use props.name to display the value.